FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Image Processing for Detection of Vehicles in Motion

Francisco José Lopes Veiga

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor Dr. Rosaldo José Fernandes Rossetti

Second Supervisor: João Francisco Carvalho Neto

September 12, 2018

Image Processing for Detection of Vehicles in Motion

Francisco José Lopes Veiga

Mestrado Integrado em Engenharia Informática e Computação

September 12, 2018

Abstract

Some problems that traffic managers face nowadays are traffic congestions in cities and highways, which are main contributors to the number of traffic-accidents and the increasing environmental impacts. Said congestions are caused, amongst other reasons, by the slow growth of the capacity of roads compared to the number of vehicles that travel in them. The development of new infrastructures is costly and upgrading existing ones causes traffic disruptions, leading to more problems. One solution to this problem is to improve the efficiency of the existing transportation systems. In order to do so, new and more efficient ways of gathering road usage data have to be developed.

Automatic traffic surveillance for vehicle detection, classification and tracking using vision-based methods has attracted a great deal of interest from researchers in the past decade. Even though many solutions have been proposed, real-time fully automatic systems need to be further developed. Systems based on Convolutional Neural Networks provide solutions for many of the present-day problems in vehicle detection, classification and tracking, outperforming other techniques in Computer Vision. Thus, the objective of this dissertation is to propose a framework for vehicle detection, classification and tracking in traffic videos that can be used in a series of traffic applications, such as the system here presented.

For vehicle detection and classification, we used the YOLOv3 model with pre-trained weights on the COCO dataset. In the next step, the results are fed into a multi-object tracking algorithm. The implemented tracker is based on the *IOU tracker* a track-by-detection approach where first an object detector is applied to a frame and second a tracker is used to associate the detections to tracks. Then a tracking zone is identified in the video image where the vehicles are tracked and counted.

We validated our approach by testing our application for vehicle counting in videos captured by traffic cameras. The results obtained were compared with a previous study using unsupervised learning methods. Our approach showed significant improvements on the tasks of detection, classification and tracking especially when there were occlusions.

YOLOv3 is a state-of-the-art object detection and classification system that can detect and classify objects in real time but it does not keep temporal information of the objects. We proposed a framework that introduces object tracking to the YOLOv3 model without worsening its performance. Our system achieves real-time detections, classification and tracking in videos which can be used in real applications. The domain of our framework is not limited to traffic systems and can be easily adapted to other applications such as people tracking.

Categories: Computing Methodologies→Artificial Intelligence→Computer Vision→Computer Vision Problems→Object Detection.

Keywords: Convolutional Neural Networks, Deep Learning, Vehicle Detection, Vehicle Tracking, Vehicle Counting, Intelligent Traffic Surveillance Systems.

Resumo

Alguns dos problemas enfrentados pelos gestores de trânsito são os engarrafamentos nos centros das cidades e autoestradas, acidentes de trânsito e impactos ambientais. Uma das razões para os engarrafamentos de trânsito é que a capacidade de as estradas cresce relativamente mais devagar em comparação com o número de veículos. O desenvolvimento de novas infraestruturas é dispendioso e o melhoramento das existentes pode causar interrupções indesejadas no trânsito, causando mais problemas. Uma solução para este problema é melhorar a eficiência dos sistemas de transporte existentes.

A vigilância automática de trânsito para deteção, classificação e seguimento de veículos usando métodos baseados em visão por computador atraiu uma grande quantidade de interesse dos investigadores. Mesmo existindo muitas soluções propostas, os sistemas totalmente automáticos de deteção, classificação e seguimento de veículos em tempo real precisam ser mais desenvolvidos. Os sistemas baseados em redes neurais convolucionais fornecem uma solução para muitos dos problemas atuais na deteção, classificação e seguimento de veículos, superando as outras técnicas de visão por computador por uma grande margem. O objetivo desta dissertação é propor uma estrutura conceptual para deteção, classificação e seguimento de veículos em vídeos de trânsito que possa ser usada em várias aplicações, e desenvolver um sistema baseado nesta estrutura para a contagem de veículos.

Para a deteção e classificação de veículos, utilizamos o modelo YOLOv3 com pesos pré-treinados no conjunto de dados COCO. De seguida, os resultados são encaminhados para um algoritmo de seguimento. O algoritmo de seguimento implementado é baseado no *IOU tracker*, que é uma abordagem de seguimento por deteção, onde primeiro um detetor de objetos é aplicado a uma imagem, e em segundo é feito um seguimento para associar as novas deteções às deteções dos frames seguintes. Em seguida, é identificada uma zona de seguimento na imagem de vídeo onde é feita a contagem dos veículos.

Avaliamos a nossa abordagem testando a aplicação para a contagem de veículos em vídeos capturados por câmeras de trânsito. Os resultados obtidos foram comparados com um estudo anterior que utilizava métodos de aprendizagem não supervisionados. A nossa abordagem mostrou melhorias significativas nas tarefas de deteção, classificação e seguimento, especialmente quando há oclusões.

O YOLOv3 é um sistema de deteção e classificação de objetos de última geração que pode detetar e classificar objetos em tempo real, mas não mantém informações temporais dos objetos. Propusemos uma estrutura conceptual que introduz o seguimento de objetos ao modelo YOLOv3 sem piorar o seu desempenho. O nosso sistema realiza deteção, classificação e seguimento em tempo real em vídeos, que pode ser usado em aplicações reais. O domínio da nossa estrutura não se limita aos sistemas de trânsito e pode ser facilmente adaptado a outras aplicações, como o seguimento de pessoas.

Categorias: Metodologias de Computação→Inteligência Artificial→Visão por Computador→Problemas

de Visão por Computador→Deteção de Objetos.

Palavras Chave: Redes Neuronais Convolucionais. Aprendizagem Profunda. Deteção de Veículos. Classificação de Veículos. Seguimento de Veículos. Contagem de Veículos. Sistemas Inteligentes de Vigilância de Trânsito.

Acknowledgements

First, I would like to thank everyone who believed in me and supported me in the decision to take a computing engineering degree.

I would like to thank my parents, Francisco Maximino Pinto Veiga and Ilda do Nascimento Lopes, my sisters, Sandra Manuela Lopes Veiga and Ana Cristina Lopes Veiga, and my nephews, Diogo and Clara Veloso for their love and friendship.

I would also like to thank my friends Torcato Magalhães, Luísa Pimentel and their two children, Afonso and Tiago Magalhães, for their friendship and support over the last five years in Porto. Especially you Pisco because you taught me to surf in these five years. I spent great moments with you surfing that I will never forget.

I could not forget all my friends, especially those who live in London, Kilas, J. Rocka, Nuno da Tuga, Toni, Paulinha and their children, because without you I would not have been able to achieve this goal.

All my colleagues at Faculty of Engineering of the University of Porto for their support and sharing of knowledge.

To Professor Rosaldo Rossetti for all the support, enthusiasm and knowledge passed on to me during the dissertation. I was able to perfect myself academically and as a person.

Finally, I would like to thank all the MIEIC professors for sharing their knowledge and the Faculty of Engineering of the University of Porto for the excellent opportunity to be part of this great institution.

Francisco José Lopes Veiga

"Be sure to put your feet in the right place,
then stand firm."

Abraham Lincoln

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations and Acronyms

AI	Artificial Intelligence
ALN	Attribute Learning Network
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DAVE	Detection and Annotation for Vehicles
DNN	Deep Neural Networks
DPM	Deformable Part Models
Fast R-CNN	Fast Regions-based Convolutional Neural Network
Faster R-CNN	Faster Regions-Convolutional Neural Network
FHD	Full High Definition
fps	frames per second
FVPN	Fast Vehicle Proposal Network
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
IOU	Intersection Over Union
ITS	Intelligent Transportation Systems
LIACC	Laboratory of Artificial Intelligence and Computer Science
mAP	mean Average Precision
ODN	Object Detection Network
R-CNN	Regions with Convolutional Neural Networks
R-FCN	Region-based Fully Convolutional Network
RNN	Recurrent Neural Networks
ROI	region of interest
RPN	Region Proposal Network
SPP	Spatial Pyramid Pooling
SVM	Support Vector Machine
UTS dataset	Urban Traffic Surveillance dataset
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Scope and Motivation

Intelligent Transportation Systems (ITS) is a popular and important research field that has received the attention of researchers in recent years. ITS is a comprehensive management and service system, which aims to provide innovative services relating to different modes of transportation management [LWM17].

Some problems traffic managers have to deal with almost every day are traffic congestions on city roads and highways, traffic-accidents and environmental impacts. One reason for traffic congestions is that roads capacity grow relatively slowly compared to the number of vehicles [SWN+16]. The global population is projected to rise by two billion over the next 25 years to 9.2 billion and the number of cars is projected to double by 2040 reaching the 2 billion mark[1]. With the increasing number of vehicles on the roads, there is a need to improve mobility, sustainable transportation, and convenience. In order to prevent future congestions on roadways, we have to increase the capacity of transportation systems, develop alternatives that increase capacity by improving the efficiency of the existing transportation systems and promote the use of public transit systems. ITS addresses ways to deal with the increase in travel demand on transportation systems using the second option [MK07].

Traffic managers use surveillance systems (Figure 1.1) to monitor and manage traffic, whether to monitor vehicles in urban areas, motorways, tunnels or for the collection of data. Conventional techniques for traffic measuring, such as inductive loops and weigh-in-motion sensors (Figure 1.2c) are expensive to install and demand traffic disruption during installation and maintenance [Kod13]. On the contrary video detection systems are a very cost-effective solution. Installation costs are low e.g. cameras can be installed on existing structures like on bridges that overpass motorways (Figure 1.2b). Visual image sets video detection systems apart from all detections systems. Real-time feedback received from video detection systems allow traffic managers to access

[1] https://www.weforum.org/agenda/2016/04/the-number-of-cars-worldwide-is-set-to-double-by-2040

Figure 1.1: Traffic surveillance/control system.

what is happening and take appropriate action. The combination of video images and intelligent video analytics provides both data and visual information. This data can be used to optimize traffic flow during busy periods, identify stalled vehicles and accidents [KWH+94].

(a) Video image processor (b) Traffic camera (c) Weigh-in-motion sensors

Figure 1.2: Example of vehicle detection and traffic surveillance technologies.

There is a variety of works in the literature based on video and image processing that employ different methodologies to detect vehicles and objects [ASAS16, BV11, YPC17]. However, Convolutional Neural Networks (CNN) applied to image processing is the current dominant Computer Vision (CV) theory especially in tasks such as object detection and classification.

1.2 Problem Statement

Automated vehicle detection, classification and tracking through video streams can be used in many traffic surveillance applications, such as vehicle counting. To build a computer vision-based intelligent surveillance system it is essential that its detection, classification and tracking capabilities are accurate and efficient. The ability to identify and track a vehicle in a video has proven to be extremely difficult and presents many challenges. Some problems can be related to image processing, such as detecting independently several vehicles in an image where some of these can be occluded temporarily or the entire time by other vehicles, making detection difficult or impossible. Other know problems in vehicle detection in video images are the weather and illumination. Traffic cameras are able to capture images by measuring ambient light in the real world so weather and illumination can influence the ability to detected vehicles in images. Vehicle

detection and tracking are two problems that go together, to be able to track a vehicle it is necessary to detect it first. However, most vehicle detection systems don't keep the temporal relation between frames.

Having this considered, the problem on focus in this dissertation is to find a solution to process video images efficiently and accurately in real time to automatically analyse traffic scenes to assist traffic managers in the tasks of traffic monitoring.

1.3 Aim and Goals

The aim of this dissertation is to develop a framework for automatic vehicle detection, classification and tracking in video images captured by a traffic camera, also implement a system for vehicle counting. Therefore, the main goals are as follow:

- Implement a server that can receive video images from a stream and feed them to a pipeline for processing;
- For the vehicle detection and classification tasks use a state-of-the-art object detector;
- Implement a multi-object tracking algorithm;
- Perform vehicle counting;
- Visualize the detections, classification, tracking and counting of the vehicles on a web application.

1.4 Document structure

The remainder of this document is organized as follow. Chapter 2 starts with the review of previous works on vehicle detection and classification using Deep Learning methodologies, followed by vehicle tracking and counting using different techniques. We then reviewed the state-of-the-art on object detection in still images and videos using CNN, and object tracking. Lastly, we present the available datasets in the literature for vehicle detection. The proposed framework is presented in Chapter 3. In Chapter 4 we present the details of the implementation of a simple intelligent traffic surveillance system as well as the results and discussion. In Chapter 5 conclusions and contributions of the dissertation are presented.

Chapter 2

Background and Literature Review

In this chapter we provide an overview on works that have a similar goal to ours, the development of a system for vehicle detection, classification and tacking using deep learning methods. Studying works from other authors may help to find a solution in order to solve the problems mentioned above. We also review the state-of-the-art on CNN applied to object detection and classification in still and video images. A brief description on multi-object tracking is also presented followed of an overview of vehicle images datasets.

2.1 Vehicle Detection in Intelligent Transportation Systems

Vehicle detection, classification and tracking are an important part of ITS. Vehicle detection using vision-based methods has attracted a great deal of attention, and significant contributions have been made to practical applications such as vehicle detection, classification, tracking, and counting, speed monitoring, identification of traffic accidents, traffic flow prediction, etc.

2.1.1 Background on Vehicle Detection Methods

Historically, there have been many approaches to vehicle detection that are based on appearance or motion features [YPC17, ST13, ASAS16]. Appearance features [THF07, MG05] such as size, colour or shape can be used to detect vehicles even in a stationary position. Each feature can have its strength or weakness and be used on its own or together with other features. Also, feature descriptors have been proposed such as Histogram of Gradients (HOG) [DT05] and Haar-like features [HK09]. Motion features are obtained on the movement of the vehicles. Most motion methods are based on background subtraction [Ziv04], Optical flow [ZJ11] and others [MJ11].

These methods have shown some weaknesses in the past. Including sensitivity to weather conditions, illumination, and occlusions (Figure 2.1). Compared with traditional feature-based methods, deep learning has better adaptability and better accuracy for detection and classification

Table 2.1: Vehicle detection on roadways issues in computer vision methods.

Issue	Details
Weather	Raindrops, fog, and snowflakes may make the images blurred, darken or dazzled.
Illumination	Cameras are sensitive to light. Global illumination changes, such as weather and daytime/night-time conditions in the scene. Local illumination changes, such as shadows or highlights of moving objects in the scene.
Occlusion	Vehicles can be partially or fully occluded by other vehicles or man-made structures, visual clutter, etc. This may introduce erroneous detections and fail to detect/track a vehicle.

tasks across different object types. Moreover, we will be focussing on state-of-the-art methods that use Deep Learning.

2.1.2 Vehicle Detection with Deep Learning

Deep Neural Networks (DNN) have been very successful in object detection [CZH16, ZHL+16] and object classification [KSH12, SZ15, SWY+15] so it's not a surprise that some research has been done in vehicle detection and classification using these networks.

In [ZLSM16, ZLS+17] the authors propose a framework Detection and Annotation for Vehicles (DAVE) for streaming video data with complex scenes. Apart from vehicle detection, this framework annotates the type, colour and viewpoint of the vehicle. DAVE is composed of two CNN. The first is a Fast Vehicle Proposal Network (FVPN) and aims to predict the bounding-boxes for all the vehicles in real-time. Afterwards, it passes all the vehicle candidates to an Attribute Learning Network (ALN) to validate the previous detections and simultaneously infer their type, colour and viewpoint. At training, both networks are optimized together by bridging the two CNN with latent data-driven knowledge while a two-stage inference is performed in the test phase. The vehicle detection and annotation model was trained on the CompCars dataset [YLLT15] and evaluated for vehicle detection on three other datasets: Urban Traffic Surveillance dataset (UTS dataset) [ZLSM16] which was collected and manually-labelled by the authors, the PASCAL VOC 2007 dataset [EVW+10] and the LISA 2010 dataset [YLLT15]. The results show that the proposed framework obtained better mean Average Precision (mAP) than other state-of-the-art object detectors that were trained on the same datasets, such as Deformable Part Models (DPM) [FGMR09], Regions with Convolutional Neural Network (R-CNN) [GDDM14], Fast Regions-based Convolutional Neural Network (Fast R-CNN) [Gir15] and Faster Regions-Convolutional Neural Network (Faster R-CNN) [RHGS17]. The major drawback of this framework is that the ALN can only process two Full High Definition (FHD) images per second on a Graphics Processing Unit (GPU), even that the FVPN predicts vehicle positions in real-time. This means that vehicle type (classification) and viewpoint are not inferred in real time and intelligent traffic surveillance applications may require annotations (vehicle classification) to be inferred in real time.

In [GCM+16] the authors proposed a method for real-time vehicle detection using a single CNN. In one evaluation this method predicts multiple vehicle candidates and their probabilities. The architecture of the proposed CNN extracts features from the input image to propose vehicles candidates. The network is composed of convolutional layers to extract features, inception modules [SWY+15], and uses a Spatial Pyramid Pooling (SPP) between convolutional layers and fully connected layers, which enables it to receive input images of different sizes being able to resize the images to a fixed size. The coordinates of the vehicles and probabilities are predicted by the fully connected layers. The proposed network was trained on the PASCAL VOC 2007 and 2012 [EVW+10] but in order to better the performance of vehicle detection, the authors labelled the images themselves. The authors used the ImageNet dataset [RDS+15] to test the proposed network. The results also show that this proposed method obtained 80.5% mAP in detecting vehicle classes which is a state-of-the-art result. This network runs on a GPU at 46 frames per second (fps) which means that it can process video streams in real time. However, there are some limitations to this network architecture the system struggles with small objects and nearby objects in groups.

Since the introduction of Faster R-CNN and its Region Proposal Network (RPN) some researchers have tried to solve the problem of real-time vehicle detection and classification using this detection network, as it shows great advantages in object detection over other networks. In [EVB17] the authors make a comparative study between AlexNet [KSH12] and Faster R-CNN for vehicle detection. In the AlexNet model the vehicle detection is based on background subtraction [Ziv04] using a Gaussian Mixture Model (GMM) and for classification a multi-class Support Vector Machine (SVM) is trained using the image features obtained from the CNN. The Faster R-CNN model is composed of two networks a Object Detection Network (ODN) and a RPN, and is based on a VGG16 network [SZ15]. The VGG16 is pre-trained on the ImageNet dataset and both the ODN and the RPN are re-trained on the PASCAL VOC 2007. The AlexNet model is able to detect 3 categories and the Faster R-CNN model is able to detect 20 categories. All classes different form car, bus or motorcycle are renamed as "unknown". The results show that the Faster R-CNN outperforms the AlexNet+GMM model.

Faster R-CNN achieves state-of-the-art performance on general object detection, however, its use in intelligent traffic surveillance applications, such as vehicle detection and classification, may perform unexpectedly. In [FBS16] the authors take a closer look at this approach in an attempt to better understand its underlying structure conducting a wide range of experiments and provided a comprehensive analysis of this model. The results show that through suitable parameter tuning and algorithm modifications the performance of this model for vehicle detection and classification can be significantly improved. Their experiments included an analysis of both training and test scale size, number of proposals, localization vs. recognition, and iterative training. The dataset used by the authors to train and test the model was the KITTI dataset [GLU12].

Vehicle detection and classification play an important role in intelligent traffic surveillance systems but for these systems to be used in real traffic surveillance scenarios, it is necessary that they also track the vehicle once a detection occurs. In [ZWY17] the authors proposed a model for vehicle detection and tracking. A Faster R-CNN model is used for vehicle detection while a

combination of Camshift [EBK+10] and Kalman filter [HH11] is proposed for vehicle tracking. To improve detection and tracking times the authors uses a multi-thread technique, while the GPU is used for the computations of vehicle detection the Central Processing Unit (CPU) is used for the computations of vehicle tracking.

Advances in vehicle detection and classification have been made since the introduction of DNN. However, for these models to be used in real traffic surveillance systems, they need to operate in real-time. Although there are works in the literature that can almost operate in real-time, there are still some improvements that can be made. Faster R-CNN models have introduced faster detections and accurate classifications with high mAP but still cannot achieve real-time vehicle detections in videos (in the literature reviewed in this work).

In Table 2.2 we present an overview of the methods of related work in vehicle detection and classification using DNN.

Table 2.2: Overview of the related work for vehicle detection.

Approach	Methods	Domain	Advantages	Disadvantages
[ZLS+17]	Two CNN	Vehicle detection and annotation	Fast vehicle detection, multi-task learning, multi-type vehicle classification and vehicle detection in video images with various illuminations and viewpoints.	Vehicle detection and verification together at test time make all process slow and difficult to train a model that has all vehicle types and colours.
[GCM+16]	One CNN with inception modules	Vehicle detection	Fast vehicle detection and able to receive input images of different sizes.	Struggles to detect small objects and nearby objects in groups.
[FBS16]	Faster R-CNN	Vehicle detection and classification	Parameter tuning of the network allows improvements in performance on vehicle detection.	Model doesn't achieve real-time vehicle detection and classification.
[ZWY17]	Faster R-CNN, Camshift and Kalman filter	Vehicle detection and tracking	Use of multi-thread programming help to speed up the model. Vehicle tracking.	Model doesn't achieve real-time vehicle detection.

2.1.3 Vehicle Tracking and Counting

Vehicle tracking plays an important role in intelligent traffic surveillance applications and vehicle safety as it helps to automatically monitor traffic e.g. vehicle counting, vehicles overtaking, vehicles changing lanes, accident detection, etc. In [ZWY17] the authors propose a vehicle detection based on Faster R-CNN and tracking using a combination of Camshift and Kalman filter. Camshift is based on a colour histogram and can track moving targets in a simple background. However, it can lose track when there are changes in illumination or occlusions. The Kalman filter uses motion information including velocity and sparse direction to predict target information in the next frame. The authors use a combination of the two algorithms to track vehicles. The detections done by the model are used to initialize the Kalman filter.

In [WGPQ15] presented an approach for real-time multi-vehicle tracking and counting from images captured from a fisheye camera that integrates the low-level feature points tracking and the higher level affinity based association.

The author [JBS14] proposes a method to track multiple objects of different types in urban traffic scenes. To detect objects background subtraction is used. An object model is built to keep the spatial information and feature points of the tracks. The model is updated through a state machine. When an occlusion occurs between objects, the previous observations are used, such as the positions and the size of occluded objects.

2.2 Object Detection with Convolution Neural Networks

Object classification and detection have had a lot of interest in the last few years and have become one of the most interesting fields in CV and Artificial Intelligence (AI). For instance, with the development of CNN architectures, the access to larger image datasets and advances in computer technology, computers can now outperform humans in object recognition tasks. CNN are a class of Neural Networks design to recognize visual patterns directly from pixel images with minimal preprocessing. CNN have been successfully used to identify faces and objects, detecting and tracking objects, and on self-driving cars.

One of the first CNN to be developed, in 1998, was LeNet-5 [LBBH98] and it was used mainly for character recognition tasks. One of the "biggest problems" with this network, at the time, was that it used very small images (32x32 pixels), so it was constrained to the availability of computer resources to process higher-resolution images. For some time researchers, even that they were aware of these networks, used feature-based approaches for object classification and detection. It wasn't until 2012, with the availability of GPU's, that CNN where used in larger images. The winners of the 2012 ImageNet challenge (CV challenge for image recognition) introduced AlexNet [KSH12] that used a CNN very similar to LeNet-5 but much deeper and trained on a very large dataset. They outperform all the prior winners of the competition that used feature-based approaches reducing the top-5 classification error from 26% to 15.3%. The second place that year didn't use a CNN variation scored around 26.2%. Every year since the winners and the top teams entering this competition used CNN models. As in 2015 the top-5 error was 3.57% (human performance is estimated to be 2-5%) and the winners used a ResNet CNN [HZRS15].

2.2.1 Object Detection in Still Images

Currently CNN do much more than classifying an image based on the predominant object in it. These models can be trained to detect different objects in an image, classify and segment them. However, some of the most successful object detection approaches are extensions of object classification models. As an example of an object detector, we could have a window of a fixed size taken from all positions of the input image and feed it to an image classifier [DT05]. Then, the classifier predicts if there is an object in the window or if it's the background. However, there is a problem with this method. It must process thousands of object candidates, which makes it

extremely slow. One solution to this problem would be to run a scanning detector instead of a classifier. First, we scan the image for object candidates with a generic detector and then run a classification algorithm on the object candidates [ADF12]. Nonetheless, this solution also presents a few problems. Objects can vary in size and can have a different aspect ratio, e.g. a person in a standing position is tall and cars viewed from the side are wider, we would have to scan with windows of many shapes. There are various methods for object detection in the literature and each one deals with these problems differently.

R-CNN [GDDM14] uses an algorithm called selective search [UvdSGS13] to scan the input image for possible objects, which generates approximately around 2000 region proposals. It then runs a CNN on these regions to extract features. These features are fed into a SVM to classify the region and to a linear regressor to better adjust the bounding box (if the object exists). Basically, this approach turns an object detection into a classification problem. This method is very intuitive but also very slow.

One of the reasons R-CNN was slow is that runs the 2000 object proposals on the CNN. The authors in [HZRS14] improved on this solution by introducing the SPP Network. This network uses a CNN to extract the feature maps from the entire input image only once, and it uses these feature maps to extract the features corresponding to the region's proposals generated by the selective search algorithm in the input image. This is done using a SPP to pool features in that section of the feature maps in the last convolutional layer. From the extracted features it generates its fix-length representations which are used to train the object classifier and the bounding box regressors.

The authors in [Gir15] proposed Fast R-CNN that is very similar to the SPP-Net, but they replaced the SPP layer with a "region of interest (ROI) pooling" layer (single-level SPP). Another difference in this network is that it uses a single softmax layer to output the class probabilities directly instead of training several SVM's to classify each object class. In this way we have just one CNN to train instead of one network and several SVM's. The performance of Fast R-CNN improved in terms of speed but there is a bottleneck still remaining which is the selective search algorithm to generate the region proposals.

Faster R-CNN [RHGS17] replaced the selective search algorithm used by Fast R-CNN with a very small CNN called RPN to generate ROI's. To handle the different size of objects and aspect ratio, it introduces the concept of **anchor boxes**. The network looks at each location in the last feature map and considers three bounding boxes and three aspect ratios. In total, we have nine boxes on which the RPN predicts the probability of being an object or background. The job of the RPN is to propose object regions and does not classify any objects. If an anchor box has an objectness score above a certain threshold, it gets forwarded as a region proposal. The region proposals are then fed into what is basically a Fast R-CNN. Altogether Faster R-CNN achieves better speeds and state-of-the-art accuracy.

The authors in [HCE+17] proposed a Region-based Fully Convolutional Network (R-FCN) that share all the convolutional computations in the entire image. The authors propose position-sensitive score maps to address a dilemma between translation-invariance in image classification and translation-variance in object detection. Each of these scores maps represent one relative

section of an object (e.g. top-left corner of an image of a car).

The authors in [RDGF15] proposed You Only Look Once (YOLO) which was a new approach to object detection. While previous works repurposes classifiers to perform object detection, YOLO frames detection as a simple regression problem to spatially separate bounding boxes and associated classes probabilities. It takes an image as input and predicts the bounding boxes and their class probabilities in one evaluation. YOLO divides each image into an SXS grid cell. Each grid cell predicts B bounding boxes, each bounding box has one box confidence score (objectness) and the box coordinates, and it predicts C conditional classes probabilities. The key difference of YOLO is that it only looks once at the images what makes it extremely fast, and can run detections in videos in real-time.

2.2.2 Object Detection in Videos

Despite the effectiveness of the object detectors in still images, these methods don't keep temporal and contextual information of the video. [KLY+16] presented the method Tubelets with CNN. Tubelets are bounding boxes sequences taken from different frames which have temporal and contextual information. The proposed framework is composed of four main components. First, it applies two object detection models to each frame where the results are treated separately for the remaining components. The second component sorts all still image scores in descending order. The detections with a high score are treated as high-confidence and the detections with a low score are suppressed and reduced to false positives. Then uses motion information such as optical flow to locally propagate detections, in case the detector misses a detection in a frame, to reduce false positives. The third component, starting with the high-confidence score detections, first a tracking algorithm is applied to obtain the tubelets and second, the tubelets are classified. In the last component, the results from the previous components are combined to obtain a final result.

In [HKP+16] proposed NMS Seq (off-line method) a method for object detection in videos and incorporated temporal information during the post-processing phase. Their method given a sequence of region proposals and their classes scores associates bounding boxes in adjacent frames using a simple overlap criterion. Then it selects the bounding box that maximizes the score. The selected boxes are then used to suppress overlapping bounding boxes in their respective frame. A re-scoring of the selected bounding boxes is done in order to boost weaker detections.

2.3 Object Tracking

Tracking-by-detection is one of the most practical and used research methods of multi-object tracking. Multi-object tracking aims at locating all targets of interest and maintaining their identity, and inferring their trajectories from images observation in videos sequences [ZWW+15]. The tracking in tracking-by-detection is divided into two steps. The first step is to apply object detection to each video frame, the second step is to associate tracks to detections. The main part of tracking is the association of detections with tracks between frames. To associate detections and tracks we need to estimate similarity or affinity between them. The simplest form of affinity is

overlap between the current frame and the previous frame. If the frame rate is high and object detection is good the new detections have the highest overlap [BES17].

The previous approach relies on motion. But there are some problems, if two objects are close to each other their tracks can merge together. The authors in [WBP17] proposed a method that integrates appearance information to improve the performance of tracking. This method was able to track objects through longer periods of occlusion. The author uses a CNN to extract appearance features and measure the similarity between tracks based on these extracted features. In [SAS17] proposed a method for tracking multiple targets using their appearance, motion, and even interactions. This model is comprised of three Recurrent Neural Network (RNN), appearance, motion and interaction. All models are combined in another RNN. This RNN is then used to compute the score similarity between tracks and detections.

2.4 Datasets for Vehicle Detection

During the work of this dissertation, one of the objectives was to create a dataset to train a model using transfer learning. We created a dataset with 750 images with different classes of vehicles but due to lack of time, we postpone the completion of the dataset to a next phase of the project. To annotate our dataset, we used an online tool [Sys18]. This tool offer supervised [1] methods for the annotation of the datasets. We collected a number of vehicle datasets (Table 2.3) that exist in the literature that can be used to create a super dataset. This will be done at a later stage, where incrementally we put together all the datasets with all different classes of vehicles. This dataset can then be used for the development of traffic surveillance applications.

2.5 Summary

In this Chapter we presented the related work for object detection and classification using CNN. In vehicle detection we observed that the most commonly used method is Faster R-CNN. This model is very accurate but can't achieve real time detections and classification in videos. In object detection two-stage models have high accuracy in detection while one-stage models achieve high fps. For object tracking we presented one of the most studied and efficient methods which is tracking-by-detection.

[1] Not working at the time, we were working on the dataset.

Table 2.3: Overview of vehicle datasets.

Dataset Name	Description	Number of Images
LISA Vehicle detection dataset [ST10]	Sequences captured at different times of the day and illumination settings: morning, evening, sunny, cloudy, etc. Different driving environments: highway and urban. Varying traffic conditions: light to dense traffic.	3 video sequences.
PASCAL VOC 2012 [EVW+10]	Large number of images for classification tasks. Classification, object detection.	2,260 vehicle images of cars, motorbike, bus, bicycle. 4,034 objects.
The Comprehensive Cars (CompCars) dataset [YLLT15]	The dataset contains data from two scenarios, including images from web-nature and surveillance-nature.	The web-nature data contains 163 car makes with 1,716 car models. There is a total of 136,726 images capturing the entire cars and 27,618 images capturing the car parts. The surveillance-nature data contains 50,000 car images captured in the front view.
Surveillance (UTS) dataset [ZLSM16]	Videos captured from different viewpoints and illumination conditions.	6 videos sequences.
Stanford Cars Dataset [KSDFF13]	Classes are typically at the level of Make, Model, Year.	The Cars dataset contains 16,185 images of 196 classes of cars.
ImageNet [RDS+15]	Labelled object image database.	347K vehicle images.
Kitti dataset [GLU12]	Autonomous vehicles driving through a mid-size city captured images of various areas using cameras and laser scanners.	The object detection and object orientation estimation benchmark consists of 7481 training images and 7518 test images, comprising a total of 80.256 labelled objects.
Udacity [Uda18]	Two Datasets. Classes: car, truck, pedestrian, street lights. The dataset includes driving in Mountain View, California and neighbouring cities during daylight conditions.	It contains over 65,000 labels across 9,423 frames collected from a Point Grey research cameras running at a full resolution of 1920x1200.
UA-DETRAC [LCD+17, WDC+15]	UA-DETRAC is a challenging real-world multi-object detection and multi-object tracking benchmark.	There are more than 140 thousand frames in the UA-DETRAC dataset and 8,250 vehicles.
COCO dataset [LMB+14]	Complex everyday scenes of common objects in their natural context.	30,366 images vehicle images.

Chapter 3

Methodological Approach

In this chapter, we describe the details of the framework proposed in this dissertation. First, we start by presenting the framework architecture design, as well as it inner pipeline. Then, we present the modules that constitute such architecture and describe the required methodologies and algorithms incorporated in each of its tasks.

3.1 Overview of the Proposed Framework

A typical traffic monitoring system consists of a network of traffic cameras, hardware, and software, which processes live video images captured from traffic scenes on roadways and transmits the extracted parameters in real-time to a control room where managers can monitor the traffic events [Kod13].

The architecture of the proposed framework is shown in Figure 3.1. The framework is divided into three different modules: Video processing; Event processing; and Visualization.

The video processing module receives a video input from a source, such as a traffic camera, an online video stream or any other video source. Each frame is then forwarded into a state-of-the-art vehicle detection and classification model. The model returns the parameters for the location, class, and confidence score for each of the vehicles detected in the frame. For vehicle tracking first, we identify ROI's in the scene. For each new vehicle detection within a ROI, we start to track its position until he leaves this region. For the other vehicles detections within a ROI, that are being tracked from previous frames, we update their tracking positions.

All the results from the previous module are sent to the event processor module where all the information is analysed to extract significant events. One subtask of the event processing module is vehicle counting. Within the ROI there is a virtual counting zone where the vehicles are counted by class type. In this module, other subtasks can be added to assist in traffic management, such as accident detection, speed monitoring, traffic density measurement or traffic flow prediction.

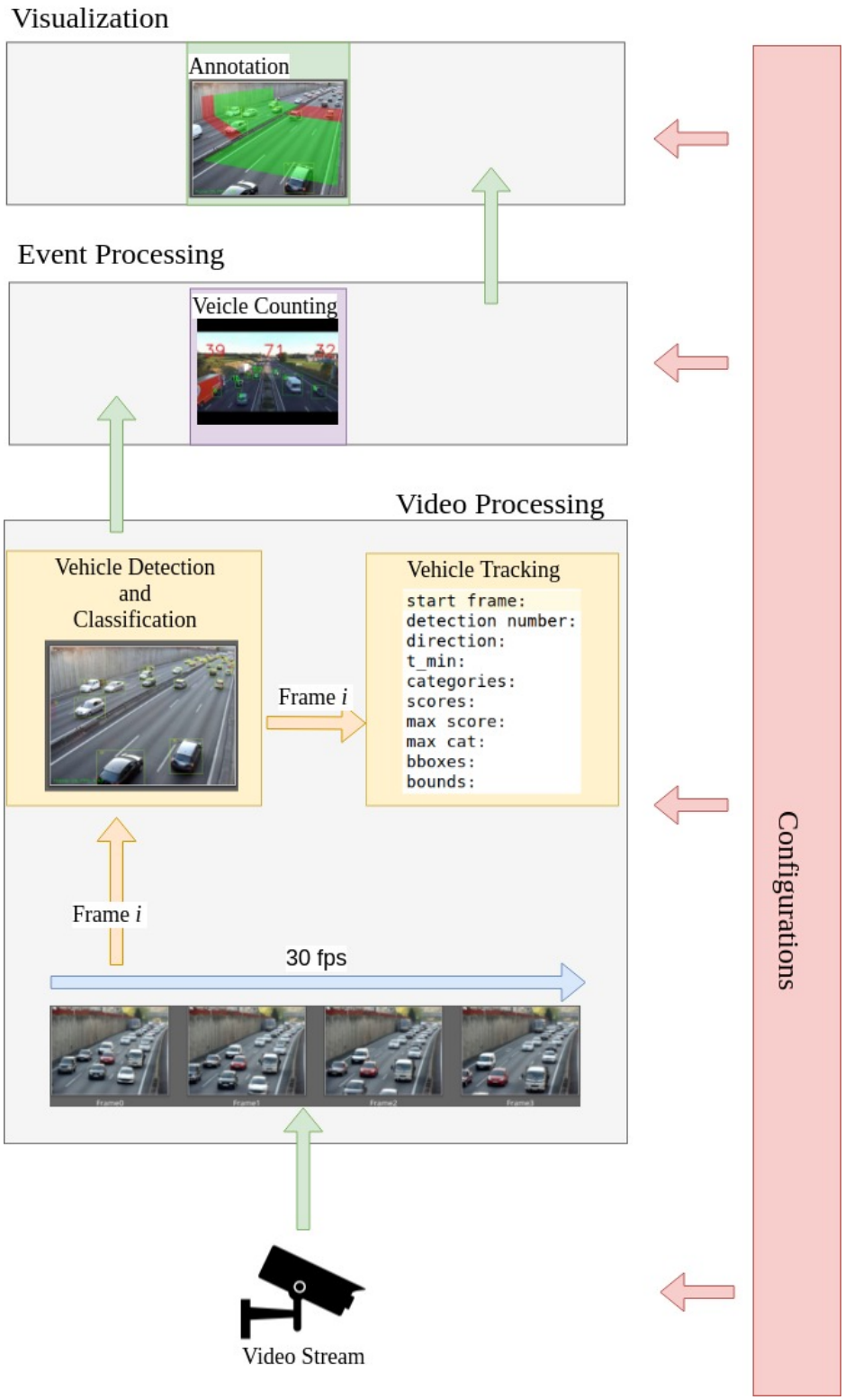
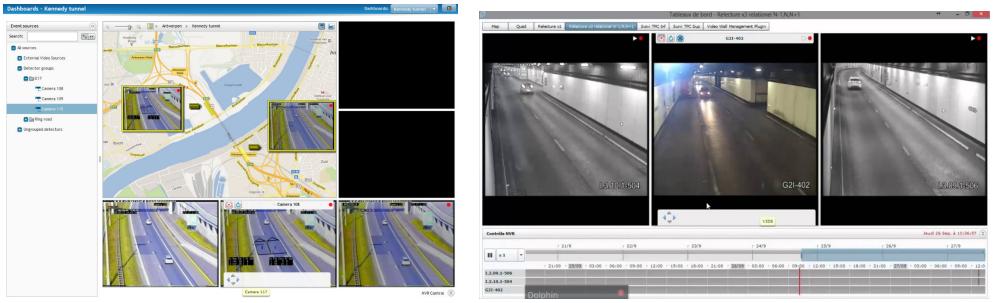


Figure 3.1: Architecture overview of the framework.

The visualization module provides an easy way to visualize the synthesized data returned by the video processing module by drawing the information in the original frames. The frames can be then streamed into a web application or dashboards. In figure 3.2 we have examples of dashboards used for traffic surveillance.

The proposed framework can be seen as a black box that receives video images from a traffic camera as input and returns the same images as output, drawing bounding boxes around the vehicles in the images, tracking their movements and keeping their counts.



(a) Dashboard set-up 1 (b) Dashboard set-up 2

Figure 3.2: Example of a dashboard of a traffic monitoring system.

3.2 Video Processing

Video processing covers most of the image processing methods but also includes methods where the temporal nature of the video data is exploited. The goal of the video processing module is to analyse video images from traffic monitoring cameras with the purpose of first finding vehicles and then extract some parameter of these vehicles, e.g. location, type of vehicle, travelling direction, etc.

In this work, we used videos captured from traffic surveillance cameras that are located on motorways. The video is then fed into the video processing module for processing. The first step is to consider the video frames as images and process those images using image processing techniques. To read the video, frame by frame, we used OpenCV [Bra00] which is an open-source computer vision library that was specially developed for video and images processing. In this work video processing can be considered as a combination of three tasks: Vehicle detection; Vehicle classification; and Vehicle tracking.

3.2.1 Vehicle Detection and Classification Model

The vehicle detection and classification tasks are the most important in this framework, as traffic surveillance applications for vehicle detection relies upon fast and accurate vehicle detection capabilities. As stated before in the literature review in the last years CNN have shown great advantages in object detection and classification. For this task in our framework, we used YOLOv3 [RF18] a state-of-the-art, real-time object detection system. This model is extremely fast and accurate and

has several advantages over classifier-based systems such as Faster R-CNN. At inference time this model looks at the whole image, so its predictions are informed by the global context in the image.

For us to be able to use YOLO in the video processing module we needed a deep learning neural network framework to implement the model. After some research in some of the frameworks available, such as TensorFlow [AAB+16], Caffe [JSD+14] and MXNet [CLL+15], we found some implementations of the YOLOv3 model that were publicly available. After running some tests to find which of the implementations delivered the best performance, in terms of fps, we chose to use the Darknet framework [Red16]. Darknet is a neural network framework written in C and Compute Unified Device Architecture (CUDA) by the authors of the YOLO model. Also, this version of YOLO (v3) is very recent and that could explain why the other implementations did not perform as well (considerable slower) as the Darknet framework.

YOLOv3 uses Darknet-53 a new 53 layer CNN for feature extraction. To run the object detection model we needed some weights for the feature extraction layers (convolutional layers). We used the weights of a pre-trained model trained on the Imagenet dataset with 1000 object classes and then fine-tuned on the COCO dataset [LMB+14] with 80 object classes.

One of the advantages of YOLO is that it makes predictions with a single network evaluation. YOLO divides the input image into a $\boldsymbol{SXS}$ grid. Each grid cell predicts only one object, e.g. in Figure 3.3 the green cells try to predict the vehicles whose centre (orange dot) fall within the same. To locate the objects, each cell predicts a fixed number $\boldsymbol{B}$ of bounding boxes and each box includes a confidence score, four parameters for the bounding box and $\boldsymbol{C}$ classes probabilities, but detects only one object regardless of the number of boxes $\boldsymbol{B}$ (Figure 3.4).

YOLOv3 uses multi-label classification meaning that same objects can have more than one label e.g. the model may label a pick-up van with two labels (car and truck). As we are using a pre-trained model, we made an adaptation of the non-maximal suppression used in YOLOv2, so we remove the duplicated labels and bounding boxes with the lowest confidence score, as we don't want to have more that one label per vehicle detected.

On the video processing module once an image is forwarded to the vehicle detection and classification model we will obtain the data of each detection as a result and not an annotated image. The data obtained from the model is the $\boldsymbol{x}$ and $\boldsymbol{y}$ coordinates of the top left corner of the bounding box, the width an the height of the bounding box, a class label and a confidence score for the class of each detection.

YOLOv3 is a state-of-the-art object detector. It's fast and accurate. It has been improved to detect smaller objects and is able to predict bounding boxes at three different scales using a similar concept to feature pyramid networks. Even using a pre-trained model on generic objects proved to be an accurate vehicle detector in images of traffic scenes.

3.2.2 Vehicle Tracking

YOLO has excellent object detection and classification capabilities, however, its design is to detect objects in a single image. At each frame repeats the same steps, detection, classification and probability scores of every object detected. Its advantage over similar object detection models is

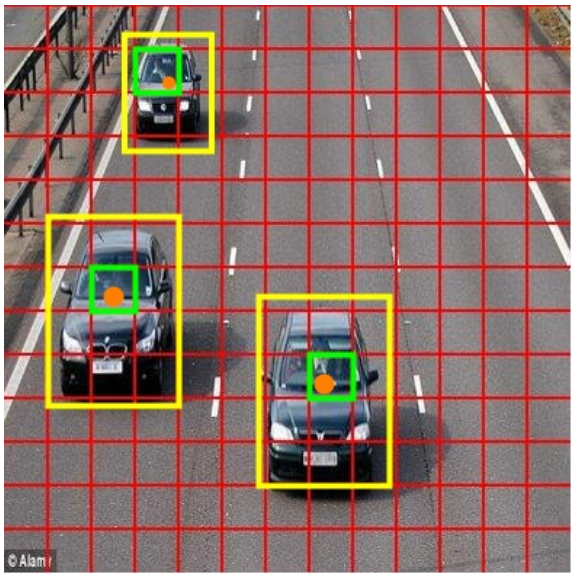


Figure 3.3: Image grid. The green grid cell is responsible for the detection of the car.

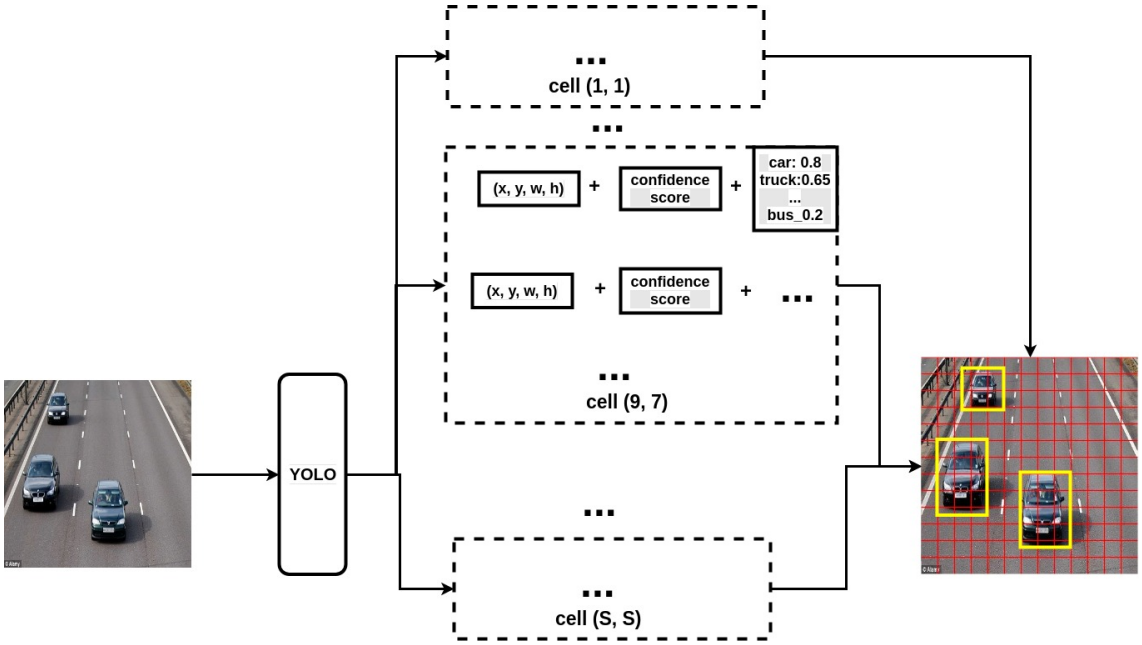


Figure 3.4: Yolov3 object detection system.

that it can repeat these steps over many frames quickly. YOLOv3 trained on the COCO dataset can process 30 fps. Even that YOLO can process many frames sequentially, it has no concept of temporal and spatial continuity between sequential frames in a video. For example, when YOLO detects an object in a frame it has no way of identifying the same object in the following frame. While there are other works in the literature that take these aspects into consideration to our knowledge they cannot achieve real-time detections in live video feeds. To overcome this issue we

introduce a post-processing phase to the YOLO model to keep temporal and spatial information of the vehicles detected by our framework.

As there are several vehicles in an image that we have to track, we have reviewed works in the literature that could do multi-object tracking. Multi-object tracking requires both an estimation of an unknown number of objects in a video and their respective path. A common approach to multi-object tracking is tracking-by-detection where first an object detector is applied to each frame and second a tracker is used to associate the detections in the video sequences. This approach suited the architecture of our framework perfectly. A typical problem for this approach is the limited performance of the underlying object detector. YOLOv3 is a state-of-the-art object detector so it should be able to handle the problems presented by this approach.

We implemented a vehicle (object) tracking system based on [BES17]. This method is based on the assumption that the detector produces a detection per frame for every object to be tracked. Furthermore, it is assumed that detections of objects in consecutive frames have high overlap Intersection Over Union (IOU) (eq. 3.1) which requires the object detection model to have high frame rates as is the case of YOLOv3.

The method proposed by the author in [BES17] starts to track all detections in a frame that weren't assigned to a track. The system will continue a track by associating the detection with highest IOU to a detection in the previous frame if a certain threshold σ_{IOU} is met (Figure 3.5). Also, all tracks that aren't assigned to a detection are kept active for a t_{min} number of frames. If the number of frames is greater than t_{min} the tracking stops.

$$IOU(a,b) = \frac{Area(a) \cap Area(b)}{Area(a) \cup Area(b)}. \tag{3.1}$$

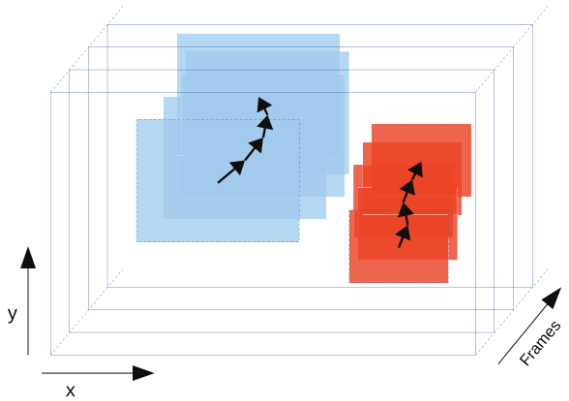


Figure 3.5: The basic principle of the IOU Tracker: with high accuracy detections at high frame rates, tracking can be done by simply associating detections with their spatial overlap between time steps [BES17].

The tracker implemented in our framework follows this principle, but we needed to make some important changes. In the method proposed by the authors, the best-unassigned detection is taken

as a candidate to extend a track. This does not necessarily lead to an optimal association between the detections and the tracks because a detection may have a higher IOU with another track that hasn't been processed yet. Our solution to this problem was to take the track as the best candidate for a detection and not the other way around (which may sound counter-intuitive). In this way, we assure that a track which object was not detected is not associated with another detection which IOU is greater than the threshold σ_{IOU}, and for every track, we guarantee the highest IOU score with a detection.

Another improvement that we have made to the tracking system described above was to predict miss detections. If a vehicle is not detected for a few frames it's possible that the IOU value may become smaller when it's detected again. We avoid this from happening at every frame by predicting the missing detections. Predicting where a bounding box should be in the next frame is not straightforward. While there are obvious options to solve similar problems, the shapes and locations of the bounding boxes returned by the detector don't follow a pattern, they are very irregular. We found a temporary but effective solution that predicts the movement of a vehicle by the movement of all points of the two last positions (eq. 3.2).

$$NewPositionBbox = LastPos + (LastPos - BeforeLastPos). \tag{3.2}$$

In Figure 3.6 we present the object life cycle based on [JBS14]. The vehicle life cycle begins when it's detected in an image for the first time. If a match is found with a new detection in the next frame, then its tracking will remain active otherwise its tracking will be lost. When an object tracking is lost for less or equal to five frames and a match is found then its tracking becomes active again otherwise its tracking ends. It may happen that the tracking is lost, but the vehicle is still in the image. In this case, the tracking of its position may start again.

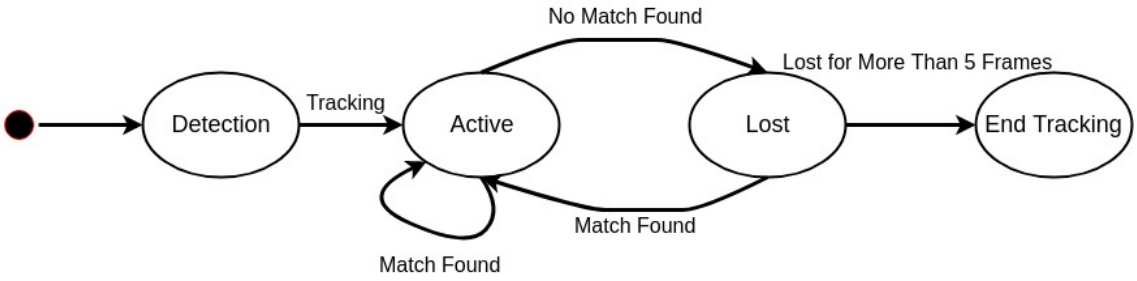


Figure 3.6: Tracking life cycle state machine.

This tracking system is very simple but it proved to be very effective and doesn't require many additional computations, most importantly it won't make the system slower. The computation cost compared to YOLO's is almost insignificant. While the object detection and classification tasks are processed by the GPU, the tracking is processed by the CPU. Once more proven to be a good match with YOLO.

3.3 Event Processing

The event processing module has the task of analysing all the data returned by the video processing module. Its function is to automatically detect any events, generate useful data related to traffic status, and create or manage alarms that might be required for the traffic surveillance/control system to operate safely and reliably. Such module plays an important role in the system as it helps to monitor and control events. In this work, we implemented the vehicle counting sub-module, but in the future other modules can be added. For example, traffic-accident detection, vehicles changing lanes detection or even vehicle travelling in the wrong direction detection.

The vehicle counting sub-module relies on the information generated by the previous module. More precisely bounding boxes coordinates, vehicle classes and travel direction of the vehicles being tracked. In order to count vehicles in video images, it was necessary to create virtual tracking and counting zones (ROI) in the video images. It is possible to configure the tracking and counting zones in these images e.g. the number of carriageways for tracking and specify the orientation of the carriageway. The virtual tracking zones in a frame can be seen in the Figure 3.7 and it corresponds to both the red and the green coloured zones. The virtual counting zones correspond to the red zones only. First, we calculate and track the centroid of the bounding box of each vehicle. When a vehicle enters the tracking zone, its status is updated to *tracking* = *True* and *counted* = *False*. Then its position is tracked on the following frames. If it reaches the counting zone, we update its status *counted* = *True*, indicating that the vehicle was counted. Vehicles are counted according to their classes. Once the vehicle leaves the counting zone the vehicle stops being tracked and we update its status to *tracking* = *False*.

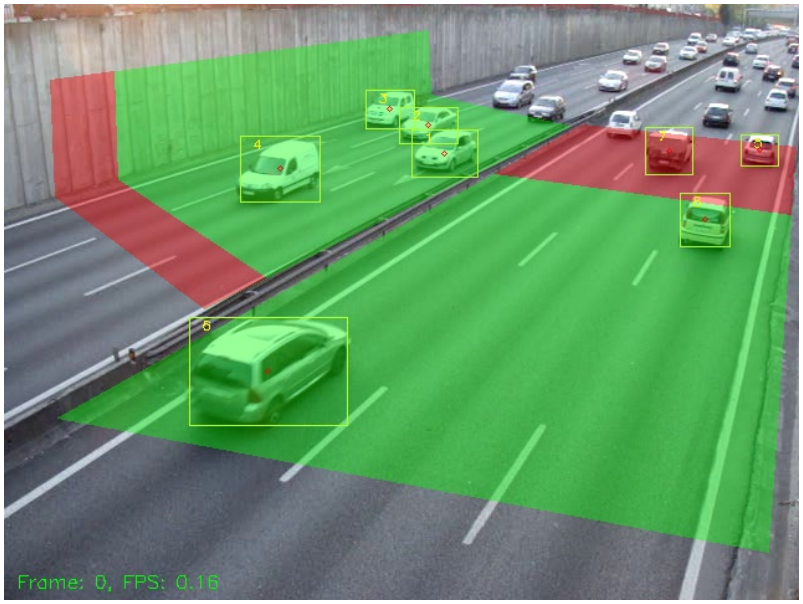


Figure 3.7: Virtual tracking and counting zone in an image.

3.4 Visualization

The visualization module is responsible to present the synthesized information produced by the other modules to the end user. The final view presented to the user is a stream of the succession of the video frames processed in the lower layers. The visualization module takes the original frames, draws the bounding boxes (different colours for each class) around the detected vehicles, frame number, frame rate and virtual tracking and detection zones. The counters are also shown to the end user, see Figure 3.8.

3.5 Summary

This chapter described the methodologies of our framework, as well as the modules that compose it. This framework is composed of three modules. The video processing module is responsible for the detection, classification, and tracking of the vehicles. The event processor module is responsible for counting vehicle, but more sub-modules can be added or even complemented. Lastly, the visualization module is responsible for the presentation of the information to the end user. All the process is done in real-time. During the implementation phase, we tried to maintain modularity in order to make it possible to make future extensions. It would be interesting to add more sub-modules to the event processor module.

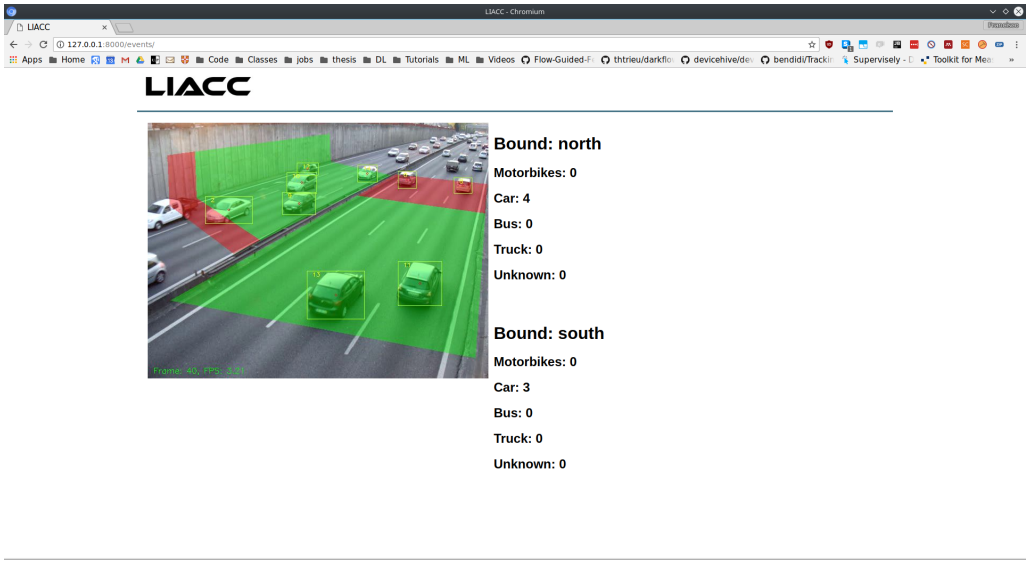


Figure 3.8: Visualization of the traffic surveillance application, classification and tracking information, virtual tracking and counting zones, and counters.

Chapter 4

Implementation and Results

In this chapter, we present the implementation of a simple intelligent traffic surveillance application according to the methods presented in chapter 3. We also performed some experiments to test our solution and present the results of the tests.

4.1 Traffic Monitoring Application

The proposed solution is to process images received from a traffic camera that monitors a section of the motorway and performs the tasks of detecting, classifying, tracking, and counting vehicles passing through that section of the motorway. The detection and classification of vehicles is done using YOLOv3 and a pre-trained model in the COCO dataset, which has 80 classes including car, truck, bus, and motorcycle. The tracking is based on the method tracking-by-detection where the detections from previous frames are used to track the objects. As a result of these processes, the application will be able to detect the vehicles and locate their exact position in the image, drawing a bounding box around them and an *id*. During the system set-up, the tracking and counting zones are superimposed onto the image. Finally, vehicles passing in the tracking and counting zones are tracked and counted. The resulting images and the counters results will be displayed in a web application so that the end user can view them.

The developed system consists of three components, as shown in the deployment diagram in Figure 4.1. The first component is the video feed output of a traffic camera server. This component transmits the video feed in real-time[1] to the video processing engine. The second component processes the video stream to detect, classify, and track vehicles. Then in each frame, it checks whether any of the detections are within the tracking and counting zones. When vehicles enter the counting zone the counters are incremented. The last step of this component is to draw all the relevant information in the frames for viewing. The annotated frames are then forwarded as

[1] For this project we used recorded videos from traffic cameras.

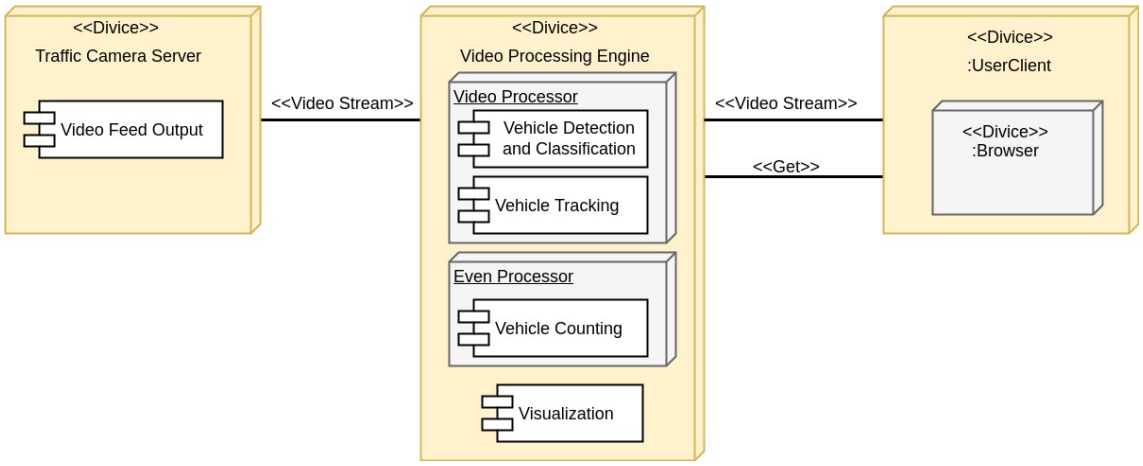


Figure 4.1: Traffic monitoring application deployment diagram.

a real-time video stream to the web application (client browser). On the client side, the counter values are updated regularly in short periods (milliseconds) with an HTTP GET request.

4.2 Video Processing Engine

The video processing engine was implemented on a Python HTTP Server based on [Dev18]. When the Server starts, it will start a thread for processing the traffic video that we want to analyse.

In Listing 4.1 the class *Deamon* inherits from *Server* a simple HTTP Server. The *cam_loop* thread runs the video processing engine. When we start the camera loop, the first task is to read a video stream s_1 from the video feed output. Before entering the *loop* to process the video frames, it starts a time counter and two variables, one for the frame number and one for counting the fps that the system is processing. Inside the *loop*, we're reading frame by frame from the video. In addition, the video event processor and visualization modules are also processed in this loop. We will review these modules in the following sections.

```
1  class Daemon is
2    input: Server S
3
4      //main variables
5      detect_frame_data ← None
6      detect_frame_data_id ← None
7      frame_id ← None
8      t1 ← None
9
10     function init is
11         initialize Server S
12         frame_id ← 0
13         create a thread t1
```

```
    end function

    function on_startup is
        start thread t_1
    end function

    function cam_loop is

        //read video
        read video stream s_1 from the video feed output

        ### start yolov3 detector ###

        //loop variables
        start_time ← time now
        frame_num ← 0
        fps ← 0

        while server S is running do

            //get video frames
            let f_i be the i^th frame read from source s_1

            ### video processor ###

            ### event processor ###

            ### visualization ###

            end_time ← time now
            fps ← fps * 0.9 + 1 / (end_time - start_time) * 0.1
            start_time ← end_time

            ### visualization ###

             //get next frame
             frame_num ← frame_num + 1
        repeat
    end function

end class
```

Listing 4.1: Pseudocode for the application server.

4.3 Vehicle Detection and Classification

For vehicle detection and classification, we used the YOLOv3 model. The source code of this project was written in Python and the source code of YOLOv3 was written in C, so a Python wrapper [mad18] was required in order to use the YOLOv3 model. YOLOv3 is used as sub-module in this project.

In Listing 4.2 when the *cam_loop* is started, before the video frames can be processed, we need to initialize the YOLOv3 detector. The detector receives three files as arguments. The first file is for the YOLOv3 network configuration, the second is the network weights trained in the object detection dataset, and the latter has additional information required for inference, such as the class names in the dataset.

Before we begin to process the video images we have to transform each image into a format YOLO understands. This step is necessary to allow the detector to process the images. Then we apply the network detector to each image. The detector returns an array of tuples, one tuple (4.1) per detection.

$$(category, score, (x, y, w, h)). \tag{4.1}$$

The YOLO model returns for each detection the type of vehicle, score relative to the certainty of the type of vehicle, and the coordinates of the bounding box. The coordinates of the bounding boxes are, x and y of the centroid position, height, and width. YOLO has a parameter that can be set in which it only returns objects detected with a confidence higher than a threshold *thresh*. Before we start doing vehicle tracking, we create a dictionary (4.2) for each detection with the data returned by the detector.

$$\begin{aligned}\{\text{"}bounds\text{"} : (x, y, w, h), \text{"}score\text{"} : score, \text{"}cat\text{"} : category,\\ \text{"}bbox\text{"} : ((x - w/2, y - h/2), (x + w/2, y + h/2)), \text{"}direction\text{"} : \_\}.\end{aligned} \tag{4.2}$$

This process is done to facilitate the data manipulation in the following steps. Besides the values contained in the tuples, we add the direction in which the vehicle is travelling. *bbox* are the coordinates of the top left and bottom right corners of the bounding box. These values (*bbox*) are used to draw the bounding box in the image.

As the YOLOv3 model uses multi-label classification there will be objects with two classes (detections). However, in our model, we want a vehicle to have only one class. To correct this problem, the *remove_duplicated* function will remove duplicates of the objects with more than one detection[2], keeping the detections with the highest confidence score.

We would like to emphasize that at this point all object classes in the COCO dataset are detected since we did not specifically train the model to detect vehicles. We will filter the detections in the processes ahead by classifying all non-vehicle objects as *"unknown"*.

[2]When an object is classified with two labels the model returns two bounding boxes for the same object.

```
class Daemon is
    ...
    //main variables
    ...
    thresh ← 0.5
    setup ← None

    function init is
        ...
        setup ← read_conf_files() //read configuration files
    end function

    function cam_loop is
        ...
        ### start yolov3 detector ###
        net ← Detector() //initialize the YOLOv3 detector
        ...

        while server S is running do
            ...
            ### video processor ###
            dark_frame ← Image(f_i) //transform frame in a darknet Image
            results ← net.detect(dark_frame, thresh) //detect vehicles in this
                frame with confidence score higher than thresh using YOLOv3
                detector
            f_results ← format_results(results, setup) //create dictionaries for
                each detection
            f_results ← remove_duplicates(f_results)
            ...
        repeat
    end function

end class
```

Listing 4.2: Pseudocode for vehicle detection and classification in the video frames.

4.4 Vehicle Tracking

In Listing 4.3 we have the pseudocode for the vehicle tracking algorithm. This algorithm runs inside the *loop* and is performed in every frame of the traffic video that we want to analyse.

For the first time that we run a frame in the detector, we will not have any active tracks, then all the detections performed in that frame will become new tracks and at the same time active tracks.

For each new track, we create a dictionary.

$$\{\text{"}bboxes\text{"}:[],\text{"}scores\text{"}:[],\text{"}start\_frame\text{"}:\_,\text{"}detection\text{"}:\_,\text{"}bounds\text{"}:[],\text{"}counted\text{"}:False,$$
$$\text{"}tracking\text{"}:False,\text{"}undetected\text{"}:False,\text{"}categories\text{"}:[],\text{"}t\_min\text{"}:0,\text{"}direction\text{"}:\_\}. \quad (4.3)$$

In 4.3 we have the dictionary with the initializations of its elements. *bboxes*, *scores*, *bounds* and *categories* are arrays that are used to keep the information of previous detections of the track. Other information that is saved is the frame number where the track started, the track number, and the direction the vehicle is travelling. Each track has a state where it can be checked whether it is within the tracking zone, it was counted or it was detected. Finally, *t_min* is the number of consecutive times a vehicle hasn't been detected.

In the following frames, if we have active tracks, then for each detection *det* we will look for their best match in the *active tracks*. If the IOU of *det* and *best_match* meet a threshold σ_{IOU}, then we have a match between *det* and *best_match* and we need to update the track with the new values of *det*.

There are three possible situations when associating tracks with detections: the first is that all detections have a track; the second is that not all detections end up with a respective track, meaning that they are new detections; the third is that not all the tracks are associated with a detection, meaning that they become undetected tracks. A vehicle may go undetected for one or more frames because the detector wasn't able to detect that vehicle. When we are left with undetected tracks, they need to be updated. The function *update_undetected_tracks* updates the status of the track to *undetected = True*. If the track goes undetected for more than a threshold *t_min* then the tracking to that vehicle is finished and we update its state to *tracking = False*. In the case where the track is undetected for less than a threshold *t_min*, we predict its position so that we don't lose its track. For the prediction, we simply calculate the difference between the two last bounding boxes in the last two frames and add it to the bounding box of the last frame. The reason for this choice was the unpredictable size of the bounding boxes for the same vehicle in consecutive frames. For the undetected tracks, we update their positions with the new predicted bounding box and its state to *t_min = t_min + 1*, *scores* with *-1* and *categories* with *undetected*. Lastly, we check if the vehicles are within the tracking area, if they are not, then we update its status to *tracking = False* and we terminate its tracking.

When looking for the best match between a track and a detection, there may be a chance that the track has no new detection for a few frames. In the case were a track has not a new detection for three or more frames and less than the threshold *t_min*, it is possible that the IOU doesn't meet the minimum of the thresholds of σ_{IOU} because the prediction of the bounding box may be slightly off its real position. In this case, we need to decrease the value of this threshold once a vehicle has been undetected three times. Lastly, when this threshold is reached, we need to update the positions of the tracks. The *updated_track* function updates the *bboxes*, *scores*, *categories*, *bounds* and *t_min = 0*, the track is removed from the active tracks and inserted in the updated tracks, and the detection is removed from the detections.

```
class Daemon(Server) is
    ...
    //main variables
    ...
    sigma_iou_lower ← 0.5
    alpha_iou_lower ← 0.3

    function cam_loop is
        ...
        //loop variables
        ...
        det ← 1
        tracks_active ← {}
        tracks_finished ← {}

        while server S is running do
            ...
            ### video processor ###
            ...
            updated_tracks = {}

            if len(tracks_active) > 0 then
                for det in detections do
                    if len(tracks_active) > 0 then
                        best_match ← track where max(iou(det, track)), track ∈
                            tracks_active
                        iou_val ← iou(det, best_match)
                        if iou_val ≥ sigma_iou_lower then
                            update_track(best_match, det, f_results, updated_tracks
                                , tracks_active)
                        else if iou_val ≥ alpha_iou_lower and best_match['
                            detection'] ≥ 3 then
                            update_track(best_match, det, f_results, updated_tracks
                                , tracks_active)
                        end if
                    end if
                repeat
             end if

             //create new tracks
             new_tracks = {}
             for det in detections do
                add det to new_tracks
                det ← det + 1
             repeat

             //update undetected
             undetected_tracks = {}
             for track in tracks_active do
```

```
46                  update_undetected_track(track, tracks_finished, undetected_tracks,
                        t_min, setup)
47              repeat
48
49              tracks_active ← updated_tracks + new_tracks + undetected_tracks
50              ...
51          repeat
52      end function
53
54  end class
```

Listing 4.3: Pseudocode of the vehicle tracking algorithm.

4.5 Vehicle Counting

Once we start to track vehicles we can perform vehicle counting. The current model used in this project is able to detect and classify 80 different classes of objects but of these classes only four are vehicles. We are only interested in the vehicle-related classes which are motorcycle, car, truck and bus, all the others are treated as "unknown". To keep count of the different classes we create five different counters, one for each class and another one for the "unknown" classes. In every frame after updating the track of the detected vehicles, we proceed to the vehicle counting (Listing 4.4).

For a vehicle to be counted its centroid needs to be within the counting zone and both its statuses of *counted* and *undetected* equal to *False*. Also, if a track status are *undetected* = *True* and *counted* = *False*, meaning that there wasn't a detection for that track in that frame, and if the vehicle is within the counting zone, it won't be counted until its status becomes *undetected* = *False*.

```
1   ...
2   ### event processor ###
3
4   if setup then
5       for track in tracks_active do
6           if track['undetected'] or track['counted'] then
7             continue
8           end if
9
10          if inside_area(track, counting_zone) then
11              track['counted'] ← True
12              idx ← get_indice(track_category)
13              if idx = 4 then
14                  update the counter for the unknown category
15              else
16                  update the counter for the category(idx)
```

```
            end if
        end if
    repeat
end if

### visualization ###
...
```

Listing 4.4: Pseudocode for vehicle counting.

4.6 Visualization

So far we have detected, classified and kept tracking of the vehicles in a sequence of frames of a video stream. For a better analysis and understanding of the data being processed visualization is required. After the detection and tracking of the vehicles to be able to visualize the relevant information we have to draw it on the frames.

The *draw_bboxes* function in the Listing 4.5 takes all active tracks and draws a bounding box and ID for each track. The colour of the bounding boxes is different for each class: motorbike - purple; car - light green; truck - dark green; bus - blue; undetected - red; unknown - black. This function also draws the tracking and counting zones if a configuration file is provided.

Before the image can be streamed into the web application we need to change its format and save it into a global variable.

Finally in the Listing 4.6 we show the controller that handles the streaming of the processed images. After the web application is started on the client browser, the handler initializes a *loop* that calls the function *get_frame* to retrieve the images from the video processing engine creating a stream. From the client side, a HTTP GET request is sent regularly to update the values of the counters.

```
...
### visualization ###

draw_bboxes(frame, tracks_active, setup) \\draw bounding boxes

# Draw additional info
end_time ← time now
fps ← fps * 0.9 + 1 / (end_time - start_time) * 0.1
start_time ← end_time

### visualization ###
frame_info ← 'Frame: frame_num, FPS: fps'
detect_frame_data_id ← frame_num
detect_frame_data ← draw_info(frame, frame_info)
...
```

Listing 4.5: Pseudocode for data visualization.

```
class Stream is
  input: BaseController BC

    fuction get is
      input: handler
        prev ← None
        while handler.server S is running do
            data, frame_id ← handler.get_frame()
            if data ≠ None and frame_id ≠ prev then
                prev ← frame_id
                handler.write(data)
            end if
        repeat
    end function
end class
```

Listing 4.6: Pseudocode for the controller of the web application.

4.7 Configuration File

When we start the server, we can provide a configuration file. In this file, we can indicate the tracking and counting zones that we want to observe in the video image. These zones are also known as ROI's. After processing an image with the detector, the *format_results* function will check if a detection is within the tracking zone, and if it's not the detection will not be processed further. When a vehicle is detected within a tracking zone for the first time, the direction of the carriageway he's travelling will be added to its track.

In figure 4.2 we have an example of a configuration file. *Bounds* is the number of carriageways covered by the image of the traffic camera that we want to analyse. *Divisions* is the fraction (*1/divisions*) of the tracking zone to be allocated to the counting zone. Where the counting zone is placed depends on the direction of the traffic. To mark the tracking zone on an image, we have to provide the points of a polygon. The tracking zone is then the area of the polygon. To get the points for the tracking zone, we use a tool found online at [Mul18]. We begin to mark the points clockwise where we want to place the beginning of the tracking zone. The *points in the counting zone* depend on the shape of the polygon. Finally, the direction of the carriageway is 0 or 1, depending on whether it's north and south, or west and east.

For this project we assume that the cameras are in a fixed position, so we have to provide the "points" of the tracking and counting zones in the configuration file.

```
number of bounds: 2

divisions: 6

bound: north
tracking area: 465,97,637,115,615,441,43,333
counting area points: 2
direction: 0

bound: south
tracking area: 161,245,41,155,37,61,341,21,343,73,455,98
counting area points: 3
direction: 1
```

Figure 4.2: Example of a configuration file.

4.8 Experiments and Results

In this section, we validated our approach by testing our application for vehicle counting. A similar work has been done previously in the Laboratory of Artificial Intelligence and Computer Science (LIACC) of the University of Porto by [NRS17] and it will be used as a comparison with our work. The methods used by the authors in [NRS17] were background subtraction and connected labelled components for the vehicle detection and fuzzy sets for the vehicle classification. The authors conducted experiments on a variety of settings that consisted of reading a video stream filmed on a motorway during a period of time and performed vehicle counting on two different classes, heavy and light vehicles. Then they compared the results of the detections performed by their algorithm and the ground truth (counting the vehicles by visual observation).

We will use three different videos with different camera set-ups (Figure 4.3) also used in [NRS17]. The videos were captured in Porto on a motorway that crosses an urban area where the camera is positioned over an overpass. The traffic intensity on the motorway is quite high, which will test our application with all kind of challenges, such as occlusions, slow and fast moving traffic and sudden changes of direction. There are also the weather and light conditions that can influence the performance of the vehicle detection and classification model. In the videos used for testing, they all have the same weather conditions which are sunny and dry, however, for lighting conditions, there are lots of shadows but the overall visibility is good.

To be able to compare our results with the results in [NRS17] we have considered only some of the classes circulating in the motorway. The vehicles that we didn't consider for counting are motorbikes, pickup trucks, transit vans (passengers and commercial) and camper vans. The reason we didn't include these vehicles has to do with the YOLO model that we use. This model has been trained on generic objects and was not necessarily trained to classify different types of vehicles. In the dataset used to train this model sometimes, pickup trucks are labelled as cars and other times as trucks, the same happens with the transit vans, sometimes they are labelled as cars, buses or trucks. Finally, we only count vehicles that are being tracked prior to entering the counting zone, if a vehicle is detected for the first time in the counting zone it won't be counted. The same goes

for wrong labelling if a vehicle entering the counting zone is labelled as "truck" and its classes is "car" then it counted as a wrong detection. We divide the other classes into light and heavy vehicles. Trucks and buses they are counted as heavy vehicles all the others are counted as light vehicles.

(a) Set-up 1 (b) Set-up 2 (c) Set-up 3

Figure 4.3: Frames from the videos used to test the application.

In the set-up 1, the camera is overlooking one carriageway, the zoom on the image is quite close what makes the vehicles larger. The view of the vehicles is almost an aerial view. The tracking zone is smaller due to the close proximity of the view angle. This view angle allows for some vehicles to be totally occluded, if a truck with a tall trailer circulates in the centre lane on this carriageway it may occlude vehicles on the right-hand lane. The traffic flow is fast in most cases and there are lots of vehicles changing lanes. The counting results from this set-up can be viewed in Table 4.1.

Table 4.1: Results from set-up 1

	Results by [NRS17]		**Our Results**	
	Light	**Heavy**	**Light**	**Heavy**
Observed	141	22	190	19
Detected	126	20	188	17
Ration	89.3%	90.9%	99%	89.5%

In the set-up 2 the camera is overlooking two carriageways and is placed on the overpass in the centre of the two carriageways, the vehicles are slightly further than in the set-up 1. The view of the vehicles is the front and the rear from a higher viewpoint. The tracking zone is bigger than the one observed before, what makes the vehicles smaller in one of the ends. If large vehicles travel in the centre lanes they may occlude smaller vehicles in the lanes to the right. There are long queues in both directions in the lanes on the right but the vehicles in the other lanes travel quite fast. Mostly the vehicles in this video travel in a straight line. The counting results from this set-up can be viewed in Table 4.2.

In the set-up 3, the camera is overlooking two bounds and is placed on the overpass to the right-hand side. The zoom on this image is closer than in set-up 2 what makes the vehicles larger. The view of the vehicles is the side front and rear from a higher viewpoint. The tracking zones are also bigger than in the set-up 1, but the vehicles in the furthest tracking area are larger than in

Table 4.2: Results from set-up 2

	Results by [NRS17]		Our Results	
	Light	**Heavy**	**Light**	**Heavy**
Observed	155	30	364	28
Detected	131	28	357	25
Ration	84.5%	93.3%	98.1%	89.3%

the set-up 2. We have the same problem with occlusions as in the previous set-up. The traffic flow is fast and there are no queues. Due to the position of the camera, there may be more occlusions from larger vehicles on the right-hand side. The counting results from this set-up can be viewed in Table 4.3.

Table 4.3: Results from set-up 3

	Results by [NRS17]		Our Results	
	Light	**Heavy**	**Light**	**Heavy**
Observed	178	28	383	26
Detected	176	24	380	15
Ration	98.9%	85.7%	99.2%	57.7%

The results obtained in the set-up 1 (Table 4.1) are positive. In total, we didn't count four vehicles. In the heavy class, the two missed counts are due to the wrong labelling of the classes, two trucks were labelled as buses. In the light class, there was a specific label for a car that was labelled as a truck and one occlusion where a car was totally occluded by a truck. In this set-up, the traffic moved quite fast and the vehicles were quite apart, also there were no queues and the tracking zone was quite close to the camera. These factors may have influenced the results.

The results obtained in the set-up 2 (Table 4.2) are positive. In the light class there were seven undetected vehicles one by total occlusion and six by being partially occluded by other vehicles. All of these occlusions happen in the furthest tracking zone where the vehicles are very small. The heavy class detected all vehicles but misclassified three has cars. This set-up showed a lot of occlusions, and it's possible if we kept counting vehicles we would have more occlusions. This set-up works well in both direction when the vehicles are separated from each other. It also showed that can handle well some occlusions but its bigger drawback is on the furthest side when small vehicle are close together in the tracking area, it has difficulty in detecting some vehicles partially occluded by other vehicles. This set-up works better for the nearest tracking zone and showed that was easier to detect larger vehicles e.g. trucks. The main reason for failing most detections was the size of the vehicles being to small and being close to other vehicles.

The results obtained in the set-up 3 (Table 4.3) are positive for the light class vehicles but surprisingly not as good for the heavy class vehicles. In the light class, we had one missed vehicle by total occlusion and two that were not detected. In the heavy class there was one vehicle that was not detected, and ten that were detected but wrongly classified, five were classified as cars and five were classified has buses. The size was not a factor in this set-up but the position of the

vehicles may have played a part in the wrong classification of the trucks.

Comparing our results with the results in [NRS17] we can conclude that we have an advantage in the detection and tracking of the light vehicles having counted almost 99% of all light class vehicles. The results for the heavy class vehicle were identical in the two-first set-ups with around 89.5% of the vehicles counted, in the last set-up our performance drop significantly to around 57%. We can explain this by understanding how deep learning networks are trained. The COCO dataset has 12,786 training images for cars, 4,141 for buses and 6,377 for trucks. The model used for this project was trained on this dataset so it detects a car better because the training set has more cars images that the other classes. For a model to have better accuracy the dataset "needs" to have the same number of images for each class. In the set-up 3, all heavy class vehicles were detected but wrong classified partially by the unusual position of the vehicles in the image. For a classifier to be more accurate the images for training the model need to cover all different angles of the object, have the object in different scenarios with different shapes and colours. Vehicles have a lot of varieties especially trucks, there may not be as many on the roads as cars but certainly, there is more variety in shapes and sizes. We also noted that this model used for vehicle detection and tracking had some difficulty in detecting the small vehicle that are occluded by others. This is a problem that can be solved by fine-tuning the parameters of the network in training and or training the network with images at different scales (YOLO already does this but we can improve the dataset).

4.9 Summary

In this chapter, we provided the steps for the implementation of a traffic surveillance application to analyse and extract information from images. For each object detected coordinates of the bounding boxes where taken and a class was predicted according to its features. Further, a ROI was selected in the image as to track and count the vehicles. The results of the experiments show that our method improves on the detection and tracking of vehicles in video sequences compared with a method used previously on the same video sequences.

Chapter 5

Conclusions and Future Work

The proposed work for this dissertation was to implement a system for automatic vehicle detection, classification and tracking in video streams in real-time. The purpose of this system is to be used in traffic surveillance applications, such as vehicle counting or accident detection. We presented a framework that describes the architecture how to build such system. The system was designed with the intention of being used on motorways, but its application covers different domains related to traffic surveillance, security, monitoring, etc.

In addition, we also implemented a system based on our framework. For the vehicle detection and classification tasks, we used YOLOv3 with a pre-trained model which was trained on the COCO dataset a generic object detection dataset with 1,000 classes. YOLOv3 is one of the latest state-of-the-art object detector systems and achieves real-time object detection in videos. One objective of this dissertation was to create a specific dataset to train a model on the YOLOv3 network. We have created a vehicle dataset with 750 images with different vehicle classes but to create a dataset (with more images) it's time consuming so we had to postpone the completion of the dataset to another iteration of the work. We also trained our own model where we used the weights of a pre-trained model on the COCO dataset and fine-tuned it with Kitti a vehicle dataset but on the test images, it performed worse than the model trained on the COCO dataset. So after some tests with the YOLOv3 model trained on the COCO dataset, we conclude that it performed well enough on vehicle detection and classification for a first phase of tests in our system.

For the vehicle tracking, we choose the approach track-by-detection which is a multi-object tracker. This works particularly well with an object detection system such as YOLOv3. In our observations, the tracker managed to track all objects even when they were highly occluded. Also, YOLO doesn't keep temporal information of the objects it detects what makes this tracker an excellent addition to YOLO. Both YOLOv3 and the tracker are very fast which makes them suitable for real-time applications.

Our system proved to be excellent at detecting, classifying and tracking vehicles even that the model used for detection was trained on a generic object detection dataset. The reason that such

model works so well is the variety of images in the training dataset that allows the model to learn the different features of the objects. In our research, we learned that to train an object detector model the dataset needs to have a large variety of images of the objects and also the number of pictures of the different classes needs to be more or less the same for the test and training images.

One advantage of our proposed systems is that it's not limited to the detection, classification and tracking of vehicles. It can be used to track any other objects including people so it can be used in other areas, such as surveillance and security.

Also, we can take advantage of the pre-trained model detecting other objects apart from vehicles e.g. people. If there are object or people in the roadway (motorway) we could raise alarms so that the operators of the surveillance systems are aware of these occurrences, so security systems could be implemented as well.

Working with the YOLOv3 model and the implemented tracker helped us understand how these systems work and we could make some improvements if we needed to. Systems like YOLO are already state-of-the-art and with appropriate datasets and fine-tuning of the network parameter we could use them in real traffic surveillance applications.

5.1 Main Contributions

The contributions of the dissertation can be split into Technological, Scientific and Applicational contributions.

- **Technological contributions** We implemented a server for the analysis of images form video streams. Our system is able to detect, classify and track vehicles. Our system is also able to count the vehicles in the images. To build the system we used and modified open-source code and implemented our own. Our code is also publicly available for the community. Also, our framework may serve as the foundation for other applications in traffic surveillance systems and other domains.

- **Scientific Contributions** To our knowledge we were the first to used YOLOv3 for a real-time traffic surveillance application. Also, we were the first to implement a framework that implemented YOLO and the IOU Tracker a track-by-detection method. We also implemented the prediction of missed detections in the IOU Tracker which the authors fail to do, and we made alterations to the original algorithm to avoid wrong matching of new detections and active tracks.

- **Applicational Contributions** From our knowledge (literature review in vehicle detection) our work is the only vehicle detection and tracking model that achieves real-time detection and tracking using deep learning models.

5.2 Future Work

During the dissertation work a number of challenges appeared that would be interesting to address.

- **Create a specific vehicle dataset to train deep learning frameworks:** If we want to use deep learning models for real traffic application it is essential to have large datasets with accurate annotations. It would be interesting to use semi-supervised or supervised techniques to create these datasets as there is a lack of specific training datasets for vehicle detection in traffic environments.
- **Train YOLOv3 on a specific vehicle dataset with different classes:** It would be interesting to train YOLOv3 on a specially crafted dataset in images of vehicles of different classes for the context of real traffic applications as the YOLOv3 shown promising results in doing these tasks.
- **Dynamic detection of the tracking zones:** In this work we considered that the traffic cameras are in a fixed position but in real life, cameras can be dynamic. The automatic detection of the tracking zones would be an interesting addition to the system.

References

[AAB+16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[ADF12] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, nov 2012.

[ASAS16] M Al-Smadi, K Abdulrahim, and R A Salam. Traffic surveillance: A review of vision based vehicle detection, recognition and tracking. *International Journal of Applied Engineering Research ISSN*, 11(1):713–726, 2016.

[BES17] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-Speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Lecce, Italy, aug 2017. IEEE.

[Bra00] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[BV11] Olivier Barnich and Marc Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, jun 2011.

[CLL+15] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[CZH16] Gong Cheng, Peicheng Zhou, and Junwei Han. Learning Rotation-Invariant Convolutional Neural Networks for Object Detection in VHR Optical Remote Sensing Images. *IEEE Transactions on Geoscience and Remote Sensing*, 54(12):7405–7415, dec 2016.

REFERENCES

[Dev18] DeviceHive. Open source iot data platform. https://github.com/devicehive/devicehive-video-analysis, 2013–2018. Visted on 16 of June 2018.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, volume I, pages 886–893, San Diego, CA, USA, USA, 2005. IEEE.

[EBK+10] David Exner, Erich Bruns, Daniel Kurz, Anselm Grundhöfer, and Oliver Bimber. Fast and robust CAMShift tracking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, pages 9–16, San Francisco, CA, USA, jun 2010. IEEE.

[EVB17] Jorge E Espinosa, Sergio A Velastin, and John W Branch. Vehicle Detection Using Alex Net and Faster R-CNN Deep Learning Models: A Comparative Study. In Halimah Badioze Zaman, Peter Robinson, Alan F Smeaton, Timothy K Shih, Sergio Velastin, Tada Terutoshi, Azizah Jaafar, and Nazlena Mohamad Ali, editors, *Advances in Visual Informatics*, pages 3–15, Cham, 2017. Springer International Publishing.

[EVW+10] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, jun 2010.

[FBS16] Quanfu Fan, Lisa Brown, and John Smith. A closer look at Faster R-CNN for vehicle detection. In *IEEE Intelligent Vehicles Symposium, Proceedings*, volume 2016-Augus, pages 124–129, Gothenburg, Sweden, jun 2016. IEEE.

[FGMR09] Pedro F Felzenszwalb, Ross B Girshick, David Mcallester, and Deva Ramanan. Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1–20, sep 2009.

[GCM+16] Xiao-Feng Gu, Zi-Wei Chen, Ting-Song Ma, Fan Li, and Long Yan. Real-Time vehicle detection and tracking using deep neural networks. In *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 167–170, Chengdu, China, dec 2016. IEEE.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587, Columbus, OH, USA, 2014. IEEE.

[Gir15] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2015 Inter, pages 1440–1448, Santiago, Chile, dec 2015. IEEE.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[HCE+17] Shawn Hershey, Sourish Chaudhuri, Daniel P.W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN architectures for large-scale audio classification. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 131–135, may 2017.

[HH11] Shengluan Huang and Jingxin Hong. Moving object tracking system based on camshift and Kalman filter. In *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 1423–1426, XianNing, China, apr 2011. IEEE.

[HK09] A. Haselhoff and A. Kummert. A vehicle detection system based on haar and triangle features. In *2009 IEEE Intelligent Vehicles Symposium*, pages 261–266, Xi'an, China, June 2009.

[HKP+16] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S. Huang. Seq-nms for video object detection. *CoRR*, abs/1602.08465, 2016.

[HZRS14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[JBS14] Jean-Philippe Jodoin, Guillaume-Alexandre Bilodeau, and Nicolas Saunier. Urban Tracker: Multiple object tracking in urban mixed traffic. In *IEEE Winter Conference on Applications of Computer Vision*, pages 885–892, Steamboat Springs, CO, USA, mar 2014. IEEE.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[KLY+16] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, and Wanli Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. *CoRR*, abs/1604.02532, 2016.

[Kod13] Lucky Kodwani. *Automatic Vehicle Detection, Tracking and Recognition of License Plate in Real Time Videos*. Masters thesis, National Institute of Technology Rourkela, 2013.

[KSDFF13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, Lake Tahoe, Nevada, USA, 2012. Curran Associates Inc.

[KWH+94] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 4, pages 3776–3781. IEEE, 1994.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

[LCD+17] Siwei Lyu, Ming-Ching Chang, Dawei Du, Longyin Wen, Honggang Qi, Yuezun Li, Yi Wei, Lipeng Ke, Tao Hu, Marco Del Coco, et al. Ua-detrac 2017: Report of avss2017 & iwt4s challenge on advanced traffic monitoring. In *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pages 1–7. IEEE, 2017.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[LWM17] Yangxin Lin, Ping Wang, and Meng Ma. Intelligent Transportation System(ITS): Concept, Challenge and Opportunity. In *Proceedings - 3rd IEEE International Conference on Big Data Security on Cloud, BigDataSecurity 2017, 3rd IEEE International Conference on High Performance and Smart Computing, HPSC 2017 and 2nd IEEE International Conference on Intelligent Data and Securit*, pages 167–172, Beijing, China, may 2017. IEEE.

[mad18] madhawav. A python wrapper on darknet. compatible with yolo v3. https://github.com/madhawav/YOLO3-4-Py, 2015–2018. Visted on 17 of June 2018.

[MG05] Xiaoxu Ma and W. E. L. Grimson. Edge-based rich representation for vehicle classification. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1185–1192 Vol. 2, Beijing, China, Oct 2005.

[MJ11] Qing Ming and Kang-Hyun Jo. Vehicle detection using tail light segmentation. In *Proceedings of 2011 6th International Forum on Strategic Technology*, volume 2, pages 729–732, Harbin, Heilongjiang, Harbin, Aug 2011.

[MK07] Luz Elena Y Mimbela and Lawrence a Klein. *Summary Of Vehicle Detection And Surveillance Technologies Used In Intelligent Transportation Systems*. Federal Highway Administration, Intelligent Transportation Systems Joint Program Office, Washington, DC, 2007.

[Mul18] Darid D. Muller. Easy imagemap generator. http://imagemap-generator.dariodomi.de/, 2018. Visted on 17 of June 2018.

[NRS17] João Neto, Rosaldo Rossetti, and Diogo Santos. *Computer-Vision-based surveillance of Intelligent Transportation Systems*. Masters degree, Porto University, 2017.

[RDGF15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

REFERENCES

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[Red16] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. Visted on 12 of June 2018.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[RHGS17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, jun 2017.

[SAS17] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *CoRR*, abs/1701.01909, 2017.

[ST10] Sayanan Sivaraman and Mohan Manubhai Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):267–276, jun 2010.

[ST13] Sayanan Sivaraman and Mohan Manubhai Trivedi. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1773–1795, dec 2013.

[SWN+16] Nilakorn Seenouvong, Ukrit Watchareeruetai, Chaiwat Nuthong, Khamphong Khongsomboon, and Noboru Ohnishi. A computer vision based vehicle detection and counting system. In *2016 8th International Conference on Knowledge and Smart Technology (KST)*, pages 224–227. IEEE, feb 2016.

[SWY+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, Boston, MA, USA, jun 2015. IEEE.

[Sys18] Deep Systems. Supervisely llc. https://supervise.ly/, 2018. Visted on 28 of June 2018.

[SZ15] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, pages 1–14, sep 2015.

[THF07] L. W. Tsai, J. W. Hsieh, and K. C. Fan. Vehicle detection using normalized color and edge map. *IEEE Transactions on Image Processing*, 16(3):850–864, March 2007.

[Uda18] Udacity. Our mission is to bring accessible, affordable, engaging, and highly effective higher education to the world. https://github.com/udacity/self-driving-car/tree/master/annotations, 2018. Visted on 28 of June 2018.

REFERENCES

[UvdSGS13] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, sep 2013.

[WBP17] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017.

[WDC+15] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *arXiv CoRR*, abs/1511.04136, 2015.

[WGPQ15] Wei Wang, Tim Gee, Jeff Price, and Hairong Qi. Real time multi-vehicle tracking and counting at intersections from a fisheye camera. In *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, pages 17–24, Waikoloa, HI, USA, jan 2015. IEEE.

[YLLT15] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 3973–3981, Boston, MA, USA, jun 2015. IEEE.

[YPC17] Zi Yang and Lilian S.C. Pun-Cheng. Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. *Image and Vision Computing*, 69:143–154, jan 2017.

[ZHL+16] Dingwen Zhang, Junwei Han, Chao Li, Jingdong Wang, and Xuelong Li. Detection of Co-salient Objects by Looking Deep and Wide. *International Journal of Computer Vision*, 120(2):215–232, nov 2016.

[Ziv04] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pages 28–31 Vol.2, Cambridge, UK, 2004. IEEE.

[ZJ11] Wei Zhan and Xiaolong Ji. Algorithm research on moving vehicles detection. *Procedia Engineering*, 15:5483 – 5487, 2011. CEIS 2011.

[ZLS+17] Yi Zhou, Li Liu, Ling Shao, Senior Member, and Matt Mellor. Fast Automatic Vehicle Annotation for Urban Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1–12, jun 2017.

[ZLSM16] Yi Zhou, Li Liu, Ling Shao, and Matt Mellor. DAVE: A unified framework for fast vehicle detection and annotation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9906 LNCS, pages 278–293, Cham, jul 2016. Springer.

[ZWW+15] Shun Zhang, Jinjun Wang, Zelun Wang, Yihong Gong, and Yuehu Liu. Multi-target tracking by learning local-to-global trajectory models. *Pattern Recognition*, 48(2):580–590, feb 2015.

[ZWY17] Yongjie Zhang, Jian Wang, and Xin Yang. Real-time vehicle detection and tracking in video based on faster R-CNN. *Journal of Physics: Conference Series*, 887(1):012068, aug 2017.